INTEGRATED DIGITAL SYSTEMS DESIGN METHODOLOGY

Ramon Risério Dourado Leite (ramon@cdpee.ufmg.br)
Diógenes Cecílio da Silva Jr (diogenes@cpdee.ufmg.br)

Universidade Federal de Minas Gerais – Departamento de Engenharia Elétrica

ABSTRACT

The emergence of system-on-chip (SoC) technology is leading to a great raise on the current electronic circuitry complexity. As a result, adapted methodologies to SoC design are being developed in order to increase productivity and fit time-to-market. In this paper, a design SoC methodology is presented – based on SystemC language. It handles four abstraction levels to describe, simulate and verify hardware systems and decrease project's development and verification time. It is presented a simple 8-bit adder case study that uses this methodology.

1. INTRODUCTION

Today, due to the continuous development of silicon process technology, complex systems can be embedded into a single circuit, named Systems-on-Chip (SoC) [2]. SoCs integrate the principal functional elements of a complete final product in a single chip. The emergence of SoC, the great complexity of the new chips and the decreasing market window is forcing fundamental changes in the way electronic systems are projected [1].

Methodologies are being created and adapted to promote higher level of project productivity [4]. Such changes of methodology aim at to reduce time and development effort, to increase previsibility and to reduce the risk involved in the project of these complex CIs.

One approach that is been consolidating as an alternative to increase the project productivity is the component reuse in projects based on pre-designed components (IP's - Intellectual Property) and in platforms [4]. Platforms are common architectures that use libraries of IP's and other tools as support technologies.

This article proposes a development methodology for IP designs, presents a case study that shows the difference between four abstraction and modeling levels of the methodology, and also indications of the benefits of your use for decreasing development and test designs time.

2. METHODOLOGY

Traditional design methodologies, such as that use hardware description languages (HDLs), start from very detailed design models [3]. Designers tend to make many mistakes due to these detailed models, causing re-design, and increasing simulation and debugging time. Therefore, due to your complexity, electronic systems design has forced the use of more abstract models.

SystemC is a class library that allows to model, to simulate and to verify hardware systems and software in several abstraction levels. It's based on C++, but extends its capacities, bringing important concepts such as concurrency and events [3].

The proposed methodology makes use of SystemC to implement top-down technique defining and testing systems components. It starts from a high level model, making possible higher simulation speed. As higher level models are tested and approved, design is detailed more and more until the RTL (Register Transfer Level). This level has simulation time with more accuracy. Then, RTL design model, written in SystemC, can be translated into a HDL, such as Verilog HDL and VHDL, to be synthesized. Testbench - which is also written in SystemC - is reused to assure that successive refinement steps of the model don't introduce errors. This methodology of analysis, simulation and test cycles are used to minimize or to eliminate risks associated to the design, thus, decreasing development time.

The methodology foresees four models: reference model - that can be described in C, C++ or SystemC; behavioral and RTL models - that must be written in SystemC; and synthesis - that can be written in Verilog or VHDL. This increasing of model and abstraction levels should be compensating by raising the productivity. These models will be presented below.

- **Reference Model:** the reference model, or functional specification, is the earliest conceptual model. It's a high level abstraction that seeks to decrease complexity to define and to test the functional aspects of the system to be designed. Therefore, it allows an increase of the simulation speed the reducing of time and debugging effort.
- **Behavioral Model:** the behavioral model introduces concept of time and it may be described using transactions [5]. Once obtained the timing model, several hardware and software architectures can be explored.
- **RTL Model:** the RTL model is synchronized by clock signs. This model is synthesizable and describes the behavior in terms of finite state machines.
- **Synthesis:** Starting from RTL synthesizable model it can be chosen two non excluding roads: to synthesize the system into a FPGA (Field Programmable Gate Array), or to implement it in a chip or ASIC.

3. ADDER CASE

For better understanding of methodology models, code pieces of an 8-bit adder design are presented. This adder must add two 8-bit integer numbers and to produce the sum result, and the overflow occurrence.

3.1. Reference Model

The code shown in Figure 1 was made in accordance to a reference model characteristics and uses SystemC. In spite of based on SystemC, the code doesn't use specialized language resources, limiting itself to the adder functional aspects - addition of two numbers – i.e., without worrying with implementation details. The simulation CPU run time of entire code of reference model to perform 225 additions was 0.05 s.

```
while (infile >> a >> b) {      // read numbers
        sum = a + b;            // sum numbers
}
```

Fig. 1 Reference Model

3.2. Behavioral Model

The code shown in Figure 2 was made in accordance to reference model characteristics and it use SystemC too. In this methodology step, specific characteristics of SystemC are used, such as concurrency and timing. It shows a process (process_adder) that is permanently being executed, i.e., it simulates real circuit operation. Other processes can be executing simultaneously.

```
void adder::process_adder( ) {
   while(true) {          // process  works permanently
      sum = a + b;        // sum values previously read
      wait(5, SC_NS);   }       // delay simulation
}
```

Fig. 2 Behavioral Model

In the behavioral model, timing is defined by estimates or imposed by design requirements. CPU run time of entire code of behavioral model to perform 225 additions was 0.08 s (60% higher than reference model execution time).

3.3. RTL Model

The code shown in Figure 3 presents the RTL model characteristics, i.e., it represents an 8-bit adder concurrent process. It must be synthesizable, therefore timing characteristics can't be simulated into the process. Timing must be simulated by clock, which is established in another code part. Clock frequency is defined by estimates or imposed by design requirements.

```
void adder_RTL::process_adder( ){
   sc_uint<9> s;                  // number sizes
   sc_uint<8> aux_a, aux_b, aux_sum;
   sc_bit aux_carry_out;
   aux_a = a.read();              // read entry ports.
   aux_b = b.read();
   s = aux_a + aux_b;             // operation
   aux_sum = s.range(7, 0);
   aux_carry_out = s[8];          // overflow
   sum.write(aux_sum);            // write output ports
   carry_out.write(aux_carry_out);       }
```

Fig. 3 RTL Model

CPU run time of entire code of behavioral model to perform 225 additions was 0.12 s (50% higher than behavioral model execution time, and 140% higher than RTL model execution time).

This code can be refined, using HDLs, such Verilog or VHDL, that allows the adder circuit to be implemented in a FPGA. It should be noticed in the code above the concern with implementation details, such as the number size and overflow (carry_out) signal.

Figure 4 presents the results produced by the adder (RTL model), using GTKwave software [6]. Six entries were used for test. Outputs attest appropriate adder operation.

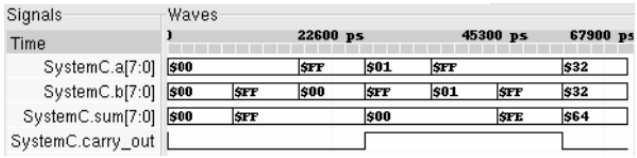

Fig. 4 GTKwave image – results of adder RTL model

4. CONCLUSIONS AND PERSPECTIVES

Based on development of a simple system design (8-bit adder), it was verified that run times for simulation grows up as model becomes more detailed. RTL model, which is commonly used as entry by traditional design methodologies presented bigger run time. This is an indicative of methodology benefit, because less detailed models usage, in addition to allow smaller simulation time, reduces significantly time debugging, increasing designer productivity.

A video controller design is being implemented using the presented methodology. Design results are reaffirming methodology benefits. As future work, RTL video controller model will be refined to allow FPGA synthesis.

5. REFERENCES

[1] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, L. Todd, *Surviving the SoC Revolution – A Guide to Platform-Based Design*, Kluwer Academic Publishers, Boston, 1999.

[2] Reinaldo A. Bergamaschi and J. Cohn, "The A to Z of SoCs", Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design, pp. 790-798, 2002.

[3] Bhasker, J., *A SystemC Primer*, Star Galaxy Publishing, USA, 2002.

[4] Black, David C., and Jack Donovan, *SystemC: from the ground up*, Kluwer Academic Publishers, Boston, 2004.

[5] Bernhard Niemann, "Using SystemC for High-Level Modeling and Refinement", SNUG Papers, Europe, 2004.

[6] GTKwave, http://www.cs.manchester.ac.uk/apt/projects/tools/gtkwave/.